LISTING OF CLAIMS

1. (Previously Presented) A memory management apparatus comprising:

a data memory which comprises a plurality of data blocks, each of which comprises a plurality of sub data blocks having a predetermined size, and when there is a request for allocating memory space of a variable size, allocates memory space in units of any one of the sub data blocks and the data blocks;

a free list memory which manages a free memory space of the data memory as an entry of a plurality of entries; and

registers that store a plurality of head location information and a plurality of tail location information of the entry, wherein a first head location information in the plurality of head location information and a second head location information are used for allocation of different byte sizes in the data memory, and the free list memory and the data memory have an equal number of entries and all of the entries of the free list memory and all of the entries of the data memory have a 1:1 corresponding relationship, wherein use of pointers between the entries of the data memory and the entries of the free list memory is unnecessary.

2. (Original) The apparatus of claim 1, further comprising:

an address translator which performs address translation between the data memory and the free list memory.

3. (Canceled)

4. (Previously Presented) The apparatus of claim 1, wherein the start address of each entry forming the data memory is determined by adding entry number x data block size (n) to data memory start address value.

5. (Previously Presented) The apparatus of claim 1, wherein the data memory has a hierarchical structure, in which the data memory contains the plurality of data blocks each having memory space of n bytes when n is a power of 2 and i and j are positive integers ($i<j$), and each data block of the plurality of data blocks comprises a first plurality of sub data blocks

each having a memory space of $\frac{n}{2^i}$ bytes and each sub data block of the first plurality of sub data blocks comprises a second plurality of sub data blocks each having a memory space of $\frac{n}{2^j}$ bytes.

6. (Previously Presented) The apparatus of claim 1, wherein each entry forming the free list memory comprises:

a plurality of bit masks each indicating whether or not a sub data block of the plurality of sub data blocks is in use; and

a first pointer which indicates one entry of the plurality of entries located immediately after another entry of the plurality of entries currently selected in the free list memory.

7. (Previously Presented) The apparatus of claim 6, wherein each entry forming the free list memory further comprises a second pointer indicating an entry located immediately before an entry currently selected in the free list memory.

8. (Previously Presented) The apparatus of claim 6, wherein according to a bit mask value of one bit mask of the plurality of bit masks, the free list memory forms an n-byte entry list capable of allocating an n-byte memory space, an $\frac{n}{2^i}$-byte entry list capable of allocating an $\frac{n}{2^i}$-byte memory space, and an $\frac{n}{2^j}$-byte entry list capable of allocating an $\frac{n}{2^j}$-byte memory space, where *i*, *j* and *n* are each a positive integer, $i<j$, and *n* is a power of 2.

9. (Previously Presented) The apparatus of claim 1, wherein when the data block is formed with X sub data blocks, the registers contain $(1 + \log_2 X)$ pointer pairs for storing the head location information and the tail location information of the at least one entry, where X is a positive integer.

10. (Original) The apparatus of claim 8, wherein when memory space of $\frac{n}{2^i}$ bytes is allocated, if there is no valid entry in the $\frac{n}{2^i}$-byte entry list, the $\frac{n}{2^{i-1}}$-byte entry list is divided and allocated as the $\frac{n}{2^i}$-byte memory space.

11. (Original) The apparatus of claim 8, wherein when memory space of $\frac{n}{2^j}$ bytes is allocated, if there is no valid entry in the $\frac{n}{2^j}$-byte entry list, the $\frac{n}{2^i}$-byte entry list is divided and allocated as the $\frac{n}{2^j}$-byte memory space.

12. (Previously Presented) The apparatus of claim 8, wherein when the $\frac{n}{2^i}$-byte memory space is deallocated, if an $\frac{n}{2^i}$-byte memory space next to the deallocated memory space in the same entry is not in use, the deallocated memory space and the memory space next to the $\frac{n}{2^i}$-byte memory space are combined and deallocated as an $\frac{n}{2^{i-1}}$-byte memory space.

13. (Previously Presented) The apparatus of claim 8, wherein when the $\frac{n}{2^j}$-byte memory space is deallocated, if an $\frac{n}{2^j}$-byte memory space next to the deallocated memory space in the same entry is not in use, the deallocated memory space and the memory space next to the $\frac{n}{2^i}$-byte memory space are combined and deallocated as an $\frac{n}{2^i}$-byte memory space.

14. (Previously Presented) A memory allocation method comprising:

in a data memory,

(a) when n is a power of 2 and i is a positive integer, if the size of a requested memory space for allocation is greater than $\frac{n}{2^i}$ bytes, allocating an $\frac{n}{2^{i-1}}$-byte memory space to a valid entry existing in an $\frac{n}{2^{i-1}}$-byte entry list managed by a free list memory;

(b) if the size of a requested memory space for allocation is equal to or less than $\frac{n}{2^i}$, allocating an $\frac{n}{2^i}$-byte memory space to a valid entry existing in an $\frac{n}{2^i}$-byte entry list managed by the free list memory, but if there is no valid entry in the $\frac{n}{2^i}$-byte entry list, dividing the $\frac{n}{2^{i-1}}$-byte entry list and allocating the divided $\frac{n}{2^{i-1}}$-byte entry list as an $\frac{n}{2^i}$-byte memory space, and

(c) storing a plurality of head location information and a plurality of tail location information, wherein a first head location information in the plurality of head location information and a second head location information are used for allocation of different byte sizes in the data memory, and the free list memory and the data memory have an equal number of entries and all of the entries of the free list memory and all of the entries of the data memory have a 1:1 corresponding relationship, wherein use of pointers between the entries of the data memory and the entries of the free list memory is unnecessary.

15. (Previously Presented) The method of claim 14, wherein step (a) comprises:

(a-1) allocating an entry corresponding to a head location of the $\frac{n}{2^{i-1}}$-byte entry list as the memory space, and setting a bit mask corresponding to the memory space to a first value which indicates that the entry is currently in use; and

(a-2) updating the head location value of the $\frac{n}{2^{i-1}}$-byte entry list with the location value of a next entry in the $\frac{n}{2^{i-1}}$-byte entry list.

16. (Previously Presented) The method of claim 14, wherein step (b) comprises:

(b-1) determining whether or not there is a valid entry in the $\frac{n}{2^i}$-byte entry list;

(b-2) if the result of the determination in step (b-1) indicates that there is a valid entry in the $\frac{n}{2^i}$-byte entry list, allocating an entry corresponding to a head location of the $\frac{n}{2^i}$-byte entry list as the memory space and setting a bit mask corresponding to the memory space to the first value;

(b-3) updating the head location value of the $\frac{n}{2^i}$-byte entry list with the location value of a next entry in the entry list to which the previous entry belongs;

(b-4) if the result of the determination in step (b-1) indicates that there is no valid entry in the $\frac{n}{2^i}$-byte entry list, allocating an entry corresponding to a head location of the $\frac{n}{2^{i-1}}$-byte entry list as the memory space and setting the bit mask corresponding to the memory space to the first value;

(b-5) adding the location value of the allocated entry at the tail location of the $\frac{n}{2^i}$-byte entry list; and

(b-6) updating the head location value of the $\frac{n}{2^{i-1}}$-byte entry list with the location value of a next entry in the entry list to which the previous entry belongs.

17. (Previously Presented) A memory deallocation method comprising:

in a data memory,

(a) when n is a power of 2 and i is a positive integer, if the size of a deallocated memory space will be greater than $\frac{n}{2^i}$ bytes, deallocating an $\frac{n}{2^{i-1}}$-byte memory space to a data memory and including an entry, corresponding to the memory space in an $\frac{n}{2^{i-1}}$-byte entry list managed by a free list memory;

(b) if the size of a deallocated memory space will be equal to or less than $\frac{n}{2^i}$ bytes, deallocating a memory space of $\frac{n}{2^i}$ bytes to the data memory and including an entry corresponding to the memory space in an $\frac{n}{2^i}$-byte entry list managed by the free list memory, but if a memory space next to the memory space in the $\frac{n}{2^{i-1}}$-byte entry list managed by the entry which manages the deallocated memory space is not in use, including an entry, which corresponds to a memory space obtained by combining the deallocated memory space and the memory space next to the memory space in the $\frac{n}{2^{i-1}}$-byte entry list, in the $\frac{n}{2^{i-1}}$-byte entry list, and

(c) storing a plurality of head location information and a plurality of tail location information, wherein a first head location information in the plurality of head location information and a second head location information are used for allocation of different byte sizes in the data memory, and the free list memory and the data memory have an equal number of entries and all of the entries of the free list memory and all of the entries of the data memory have a 1:1 corresponding relationship, wherein use of pointers between the entries of the data memory and the entries of the free list memory is unnecessary.

18. (Previously Presented) The method of claim 17, wherein step (a) comprises:

(a-1) setting the bit mask of a data block corresponding to the deallocated memory space, to a first value indicating that the entry is not currently in use, and setting the location value of a next memory space to be used, to a second value indicating that there is no valid entry; and

(a-2) adding the deallocated memory space to a tail location of the $\frac{n}{2^{i-1}}$-byte entry list.

19. (Previously Presented) The method of claim 17, wherein step (b) comprises:

(b-1) determining whether or not $\frac{n}{2^i}$-byte sub data blocks included in an entry to which the deallocated memory space belongs are all in use;

(b-2) if the result of the determination in step (b-1) indicates that all of the sub data blocks are in use, setting the bit mask of the sub data block corresponding to the deallocated memory space to the first value, and setting the location value of a next memory space to be used to the second value;

(b-3) adding the deallocated memory space to a tail location of the $\frac{n}{2^i}$-byte entry list;

(b-4) if the result of the determination in step (b-1) indicates that not all of the sub data blocks are in use, deleting the entry corresponding to the deallocated memory space from the $\frac{n}{2^i}$-byte entry list;

(b-5) setting the bit mask of the data block containing the deallocated memory space to the first value and setting the location value of a next memory space to be used to the second value; and

(b-6) adding the deallocated memory space to a tail location of the $\frac{n}{2^{i-1}}$-byte entry list.

20. (Original) A computer readable medium having embodied thereon a computer program for the method of any one of claims 14, 15, 16, 17, 18 and 19.